感谢大家前来聆听本会场的最后一个讲座 这一天确实比较漫长

Alright, thank everyone for being here for the last talk of the session. It’s been a long day.

在本讲座中 我将向大家介绍Ouroboros 一个可证明安全的权益证明协议

So, in this talk, I'll tell you about Ouroboros, a probably secure proof of stake protocol.

这个工作是由我和Alexander Russell、Bernardo David和Roman Oliynykov一起完成的

And this is a joint work with Alexander Russell, Bernardo David, and Roman Oliynykov.

既然大家来到本会场听讲座 应该已经了解我们想实现的账本目标了

So, you heard in this session and what about the ledger objective.

我们的目标是构造一个协议 实现一个具有鲁棒性的交易账本

So, the goal is to construct this protocol that realizes this robust transaction ledger.

这一目标是由Juan Garay、我、以及Nikos Leonardos在论文中提出的 论文缩写为GKL

And as you've learned, this was the objective of a paper by Juan Garay, myself, and Nikos Leonardos in GKL.

我们给出了形式化定义 这就是我们的目标

We formalized this definition. This is an objective.

这也引出了其它一些后续工作 这些工作对模型进行了精炼 给出了更好的定义

And this gave rise some follow-up work, which refine the model and definition,

一些工作也出现在本会场前面的讲座中

some of which were also heard in this session.

我们现在可以认为 我们最终得到了一个基于仿真的、可组合的目标定义

With now, we can say we have finally a simulation based definition which is composable.

一旦有了设计目标 有了很明确的方向 我们就会问这样一个问题：

Now, once you have an objective, which is clearly define a set, now you can ask,

我们是否知道实现这一目标的最佳方法？

do we know the best way to realize it?

而本会场最后这个讲座的目标就是要回答这个问题

So, trying to answer that question is the topic of this final talk.

我们知道 可以通过比特币等其它我所引用的前人工作所提到的技术来实现账本目标

So, we know that the ledger can be realized by Bitcoin based on all this wonderful previous works as I cited.

但我们也知道 在现实生活中执行的协议效果告诉我们

But we also know from the way the protocol actually works in the real world that

目前实现的账本目标具有扩展性差、能源消耗多等缺点

it exhibits significant scalability and energy efficiency disadvantages.

我们是否可以用一种更高效的方法实现账本目标？

So, can we realize it in a more efficient way?

至少到目前为止 我们有一个称为权益证明的思想

So, at least at this point, that an idea that is called proof of stake,

比特币社区早在2011年就提出了这个想法

were circulated in the Bitcoin community, as early on as 2011,

这可能是一种设计高效账本记录方式的方法

as a possible way of designing a more efficient ledger.

权益证明的基本思想和技术背景是什么呢？

So, what's the background behind proof of stake?

前面其它学者的讲座中也强调 可以认为比特币执行过程的核心目标是实现选举过程

So, as also highlighted in the previous talk, you can think that at the core of bitcoins operation is an election process.

下一个区块是通过选举来产生的

So, the next block is produced in the form of an election.

某个协议参与方 也就是矿工 会被选择上 有权利对链进行扩展

So, somehow a miner, one of the protocol participants, is selected to extend the chain.

被选择上的概率正比于参与方计算哈希值的能力

And this is done with probability proportional to its hashing power.

同时 全网通过一定的规则来解决碰撞问题

And collisions that may occur are supposed to be resolved by following some rule,

一般使用最长链规则或最大难度链规则

let's say the longest chain rule or the most difficult chain rule.

权益证明背后也遵从类似的思想

So, proof of stake, the idea behind it is to follow a similar approach,

但权益证明使用实体所拥有的权益来完成选取 个体拥有的权益同样体现在账本上

but use the amount of stake that an entity has, as that is reflected in the ledger,

权益证明将个体的权益考虑其中 通过随机化过程选举出下一个扩展链的实体

and use a randomized process that takes into account that stake to elect the next entity, the next participant who will extend the protocol.

从某种角度上看 这一过程的基本逻辑也反应出当前现实生活中协议的执行逻辑

So, in some sense this respects the fundamental logic behind how these protocols operate,

因为在现实生活中 我们可以用资金来购买哈希计算能力

because after all in the real world, you can use money to buy hashing power,

通过投入更多的资金 你可以提高在系统中被选择上的概率

and as a result, you increase your probability of being elected by investing more money in the equipment of the system.

但权益证明从某种程度上看舍弃了协议执行过程中物理层面的印记

But in this case, it is somehow cutting out this physical footprint of the protocol,

直接使用区块链本身记录的权益作为实体选举过程的参考 用于对区块链进行扩展

and use directly the stake as it is reported in the blockchain itself as the main guidance for how the election process should go to elect the next entity to extend the blockchain.

我们可以按照我们感兴趣的两个维度 对所有的协议进行分类

So, you can classify all these protocols in this two dimensional space that cover two parameters of interest.

（@译者注：这里作者称x轴表示性能或消耗能量 应该为口误）

图中y轴表示执行协议的性能或消耗的能量高低 越大表示性能越好 能量消耗越低

So, you can think of the y-axis as showing how performance or energy efficiency scales.

x轴表示协议支持的去中心化程度

And on the x-axis, we have decentralization.

当同时考虑这两个因素时 我们可以认为我们的目标是实现一个账本

So, if you put it here together, and you can think the objective is to have a ledger,

一方面我们已经有了中心化的数据库 很容易知道此数据库位于图中的左上角

well we have a centralized database on the one side, and you can easily see this is on the upper left corner.

我们也有比特币 它位于图中的右下角 更去中心化 但能源消耗很大

And then we have Bitcoin, which is on the lower right, more decentralized, but non energy efficient.

还可以直接将其它业务场景下的协商协议引入账本

And there's a direct way of also implementing a ledger by just composing sequentially a series of business in agreement protocols.

我们可以看到 这种账本与比特币相比更加中心化

And you can see that this would not be as the centralized as Bitcoin is,

并且效率比中心化数据库低 毕竟在中心化数据库中 单一节点失效会使得全节点失效

and it would be less efficient of course than a centralized database which is just has a single point of failure.

如果我们可以设计出权益证明协议 至少我们认为其有希望位于图中的右上角

So, a proof of stake protocol, at least, that's the hope, if we can design it, would actually fill this much coveted space, which is here on the upper right corner,

因为权益证明协议会比通过协商协议实现的账本更加去中心化

because it can be more decentralized than this classical disagreement sequential composition,

因为权益的分布情况会随时间的变化而变化

because the stakeholder distribution may shift over time.

但同时 协议可以实现每一次心跳都生成一个区块

But at the same time, the protocol can actually produce one block every heartbeat.

如果执行协议的参与方变得特别多 协议的执行结果也不会变得太过糟糕

And this can be done without actually suffering a penalty for scaling the protocol to a large number of participants.

如果能实现这样一个协议 我们就既可以做到去中心化 由可以做到协议高效执行

So, if such a protocol could be designed, it could possibly be more decentralized and more efficient than what we have.

在最早期刚提出权益证明思想的时候 人们认为权益证明有可能达成上述目的

So, this was realized early on, when proof of stake was suggested as a possible course of action.

但很快 人们就意识到 设计权益证明协议会遇到很严重的挑战

And very soon enough, people realized that there were significant challenges that are specific to designing proof of stake protocols.

第一个挑战是存在所谓的“粉碎攻击”

First one, which has been called as grinding attacks,

因为选举过程成为了一个可计算协议 并且选取过程也会反应在区块链上

is the fact that the adversary exactly because now the election process becomes a computational protocol, which is reflected in the blockchain itself,

与比特币工作量证明协议相比 攻击者可以绕过协议的理想执行过程

it stops being this idealized process that we can think of in the blockchain protocol like Bitcoin that uses proof of work.

部分实体可能利用其计算能力使协议的执行结果发生偏斜 使结果对自己有利

So, it is possible somehow the part is to use computational power and try to bias the protocol to their advantage.

所有基于权益证明的区块链协议都会面临粉碎攻击问题

And this type of grinding attacks were present in all the protocols that were suggesting proof of stake based blockchains.

另一个被广泛研究的权益证明协议攻击方法称为“无权益风险”攻击

Another attack which has been considered and has been seen in many cases was the Nothing-at-stake attack,

这一攻击反应的事实是 权益拥有者没必要严格按照协议对指定的链进行扩展

that somehow reflected the fact that a stakeholder didn't have to commit to extending a specific blockchain,

权益拥有者可以尝试同时扩展多个链

but you could just try multiple ones at the same time.

最后 权益证明还面临一个循环的问题

And finally there was the circularity argument that

从某种角度看 我们在区块链上执行一个协议 产生随机量

somehow you would use the blockchain to run a protocol to produce randomness,

但与此同时 协议的安全性也就不得不依赖于区块链

but at the same time, now the security of the protocol would have to rely on the blockchain.

因此 如果要证明协议的安全性 我们要绕开随机量生成与区块链安全性的约束关系

So, there is a circularity between the random generation and the security of the blockchain that you have somehow to avoid if you hope to prove this protocol secure.

这就是我们的境地

So, this is where we stand.

现在的问题是 是否可以只使用权益证明思想实现我们期望的账本目标？

And the question is, is it possible to realize the ledger objective by using just proof-of-stake discipline.

我们简单论述一下协议 假定全网是同步的

So, let's recap. We are in a synchronous setting.

时间按轮数进行划分 每一轮称为一个时间槽

Time is divided in rounds, which we call them slots,

网络中的每一条消息都可以在全网扩散 发送到每一个节点

and messages are sent through a diffusion mechanism.

在前面的两个讲座中 大家应该也看到过与之相同的网络模型

So, basically, this is the same model that you've seen the previous two talks.

攻击者非常激进 这意味着攻击者掌握了很多攻击能力

The adversary is rushing, which means that he has the advantage.

他可以传递消息、篡改消息、在消息中注入自己想要的内容、重排列消息接收顺序

He can deliver messages, and spoof their source, inject messages of its own, reordered messages,

可以将一些消息只发送给部分参与方 可以拒绝将自己接收的消息发送给别人

and send some messages to some parties, and refuse some of their messages to others.

在这样的场景下 我们首先来看看我们协议的执行过程

So, in this setting, this is how a first wondering our protocol works.

这是协议执行的第一步 此时我们需要假设权益是静态的

So, this is the first stage of the protocol, where basically let's assume that the stake is static.

我们拥有固定量级的权益拥有者 每一位权益拥有者的权益都在最开始时初始化好

So, we have a fixed set of stakeholders, which are identified here at the beginning.

每一位权益拥有者都拥有特定的权益量

It's one of them with a certain amount of stake.

权益量是预定义好的 权益拥有者的权益量写在创始区块中

And that's predetermined. It's written in the genesis block.

创世区块还包含一个种子 可以认为这个种子是一个随机字符串

Now, the genesis block also has a seed, which even think of it as a random string

种子可以为每一个时间槽随机选择并指派一个所谓的领导者

that is going to produce a sequence of random choices for leaders, so called, that will be assigned to each every slot that is coming up next.

这个领导者将负责在特定的时间槽内生成区块

So, this leader here will be responsible for producing a block for that particular slot.

对于每一个时间槽 只有一个实体负责生成区块

And only that entity is going to be responsible for producing that block here.

由于种子包含在创始区块中 因此系统已经确定好每一个时间槽所对应的领导者了

Now, observe that because the seed is part of the genesis block, it's agreed already what is the slot leader in the sequence.

时间槽领导者的采样权重源于权益拥有者所拥有的权益分布

Now, the sampling of that slot leader sequence is done according to weight that is based on the stakeholder distribution,

这意味着拥有的权益越多 权益拥有者越有可能被选为特定时间槽的领导者

that means that the more stake you have, the more likely you are to be assigned on a certain slot.

除此之外 整个选举过程是独立随机的

Otherwise, the assignment is completely independent.

现在 协议开始执行

So now, the protocol starts.

所有权益拥有者拥有一个公钥和一个私钥 用于生成数字签名

All stakeholders have a public key and a secret key that corresponds to a digital signature.

现在 对于每一个时间槽 协议的执行过程都非常简单

Now, the protocol follows an extremely simple discipline on every slot.

每一个时间槽都指定了一个领导者 这个领导者将生成一个区块

There is one slot leader that is responsible, and that one is going to issue a block.

但是 实际中实体并不会严格执行此协议

It's not necessary that this may happen.

例如 某个时间槽领导者可能处于离线状态 他也可能被攻击者所控制 拒绝执行协议

For example, a slot leader might be offline, or it might be controlled by the adversary and prefer not to do that.

此种情况下 整个协议仍然继续执行 参与方会根据最长链规则扩展前一个区块

So, the protocol will still continue, and parties will extend the previous block they know following a longest chain rule.

协议非常简单 和比特币基本完全一致 但参与方不需要在解决困难问题了

Very simple, exactly like in the case of Bitcoin, excluding the fact that there is no difficulty to take into account.

整个协议非常简单 但如何分析这个协议呢？

So, the protocol is simple. But how do you analyze that?

幻灯片给出了我们对协议的安全性分析方法

So, here is the first glimpse of how do we perform the security analysis of this.

我们需要了解的第一个重要目标是所谓的“特征字符串”

So, the first important object to observe here is what we call a characteristic string.

特征字符串的长度与系统执行过程涉及到的时间槽数量一致

The characteristic string is as long as the number of slots you have in a system execution.

当时间槽归属诚实实体 则对应字符为0 如果归属恶意实体 则对应字符为1

And it's 0 if the slot belongs to an honest party, where if it's malicious, it's 1.

注意到参与方本身无法知道特征字符串的取值

Observe the characteristic string is something that the parties themselves do not know.

但我们可以观察协议执行过程和攻击者的执行策略获取特征字符串的取值

But we know that we observe the execution of the protocol that follows some adversarial strategy that is determined as the adversary proceeds together with the protocol that the parties run the protocol.

假定幻灯片上给出的就是协议执行过程中生成的特征字符串

So, let's imagine that this is the string that characterizes how the protocol executes.

我们来看看协议是如何执行的

And let's see how does the protocol extends.

根据协议执行过程 我们可以定义一个区块树 我们称之为“分叉”

And along this protocol execution, I will define this tree, which we're gonna call a fork.

可以通过特征字符串生成这一区块树

And it is produced based on that characteristic string.

最开始的特征字符为0 对应的是创始区块 我们假定创始区块的参与方总是诚实的

So, here is a 0 node, which corresponds to the genesis block. And that's always assumed to be honest.

现在 我们有了第一个时间槽 一个诚实的参与方生成了区块1

And now, we are here at the first slot, where it's an honest party that produces block 1.

接下来 我们看到了一个关联了攻击者的时间槽

Now, what happens next is that we do have an adversarial slot.

不失一般性 攻击者不一定非要在轮到他操作的一瞬间就采取行动

Without loss of generality, the adversary need not act at that particular moment.

攻击者可以等待 当时间槽将要切换到诚实参与方操作的时候再采取行动

So, he can wait. He can wait up to the third slot where we have an honest party.

攻击者可以执行下述攻击行为

And here is what the adversary does.

控制时间槽2的攻击者将产生一个区块 使得诚实参与方将在3的位置生成区块

The adversary who controls slot 2, he produces a block and presents that first to the party that will produce the slot of the 3rd position.

现在 区块链将会分为两个分支

So now, what happens is that this party and this party have been split in two.

因此 这个诚实参与方将会沿着上面的分支进行扩展 我们称此现象为一个尖叉

So, this honest party now is extending this particular path that we call a tine.

整个攻击过程会导致区块链出现3个分支

So, this particular fork, as we will see, is going to have 3 times.

而攻击者可以规划整个攻击方式 他可以强制沿着中间分支扩展区块链

And the adversary is going to schedule its attack so that it will divide the change that the parties are actually extending as the protocol proceeded.

我们来看看位置5所发生的事情

So, let's look at what happens at position 5.

在这一时刻 攻击者控制了时间槽4 他在下方生成区块 并告知时间槽5关联的参与方

At this moment, the adversary who is in control of the 4th slot, he produces a block here and presents that to party 5.

注意到此参与方会发现 包含创始区块在内 中间和下方两个分支的长度都为3

Now, observe that this party observes this time which is of length 3 including the genesis block. And it's as good as this one.

由于攻击者可以控制消息的发送顺序 他可以说服参与方在5的位置扩展区块链

The adversary who is internal control of the ordering of messages can convince that party, a position 5 to extend this one.

攻击者随后通过类似的方法在6的位置扩展区块链

In a similar way, the same thing happens in 6.

最后 攻击者舒服参与方在9的位置生成新的区块

And finally the adversary convinces the party at position 9 to extend that particular block.

我们观察到的结果是 前面所描述的简单协议 其实际执行结果可能非常复杂

So, what you have witnessed is how this simple protocol that I described in the previous slide gives rise to a rather complex execution

这是因为攻击者可以控制诚实参与方 使其对区块链上不同的位置进行扩展

where the adversary can switch the positions of the honest parties to different sides.

这会出现什么问题呢？

So, what is the problem here?

从工作量证明开始 我们已经广泛研究了共识机制 该如何从共识机制角度考虑此协议？

How can we think about this protocol with all the knowledge that we have gained through all these previous works that started on proof-of-work?

正如我们所看到的 我们面临的情况和工作量证明不太一样

The situation is quite different as we'll see.

首先 攻击者的攻击难度会大幅降低

First of all, observe that the adversary now is in much better position.

与基于工作量证明的协议执行过程相比 攻击者更容易实施攻击

So, the protocol execution, compared to proof of work protocol based execution, is much better here.

为什么？因为攻击者可以事先知道哪个权益拥有者将会被激活

Why? Because it can see ahead of time how stakeholders are activated.

攻击者可以在不付出任何代价的前提下 在相同的时间槽中创建多个不同的区块

It can generate multiple different blocks for the same slot at any time without cost.

这与比特币的执行机制不同 在比特币中 攻击者需要投入计算资源来扩展区块链

So, contrary to how Bitcoin works, where the other has to invest computational power to extend the specific like chain.

但现在情况不一样了

The same is not true here.

最后 攻击者可以等到快到诚实节点行动的时候再采取行动

Finally, the adversary can wait and act just before an honest party becomes activated.

当时间槽指向攻击者自己时 他完全可以先等待 不采取任何行动

It doesn't need to do anything in the slots that it's assigned to himself.

他可以等到诚实节点激活的时候再采取行动 他可以根据情况采取最佳攻击策略

He can just wait until an honest part is activated. And then, at that moment, he can choose the best possible course of action.

因此 攻击者不需要提前告知其想如何扩展区块链

So, there's no need for the adversary to commit ahead of time how it extends a chain.

我们来观察一下特征字符串 幻灯片给出了两个特征字符串

So, let's look at the strings and say here is the characteristic strings.

在协议实际执行过程中 特征字符串的可能性有指数多个

They are exponentially many of them in an execution.

攻击者更喜欢看到哪个特征字符串呢？

Which form of one are the ones that the adversary prefers?

大家可以看到 很明显攻击者肯定不喜欢结果为000的特征字符串

Now, you can see that the string 000 is obviously a string that the adversary does not like.

攻击者无法对这种特征字符串对应的区块链实施任何攻击 区块按顺序组织 没有分叉

And it's very easy to see that there is nothing really the adversary can do here. We have just a complete sequence like that.

另一方面 看看这个特征字符串00110001

On the other hand, just look at that string 00110001.

攻击者完全可以对这个特征字符串对应的区块链实施攻击 我们称其为可分叉的

Now, that string is actually a string that the adversary can do complete attack. We call that string forkable.

攻击者如何实施攻击呢？

How does the attack works?

大家可以看看 为何我们称此字符串是可分叉的呢？

Now, you can see it here why we call that string forkable?

仔细看看当前区块链的状态 如果你是下一个被激活的诚实参与方 你该如何行动？

Just look exactly about what happens here, and imagining that you are the next honest party to be activated exactly after this point.

我们来看看会发生什么

Now, observe how this looks.

大家可以看到 攻击者有明确的自主权 决定产生分叉的时间上限和时间下限

And just see that it is absolutely in the discretion of the adversary to serve you either the upper time or the lower time of that fork.

这是因为上方分叉的长度为5 下方分叉的长度为4 但是攻击者控制着此时间槽

That's because that time is of length 5, and this one is at length 4. But the adversary is in control of that slot.

因此 当你作为诚实参与方生成区块时 你沿着哪个分叉进行扩展完全由攻击者决定

So, when you come up here as an honest party, it will be up to the adversary to serve you whatever time it prefers.

这种情况下攻击者可以成功实施攻击 此特征字符串比较糟糕 我们称其为可分叉

So, this is a bad string for us in a string that the adversary wins. So, we call that string forkable.

这里的核心点在于 如果我们随机选择一个特征字符串 其可分叉的概率有多大？

So, the main point is going to be, if you pick a string at random, what is the probability that it's forkable?

这就是安全性分析的核心点

And that's the core point of the security analysis.

为了明白这一点 我们需要引入两个很有帮助的参数

To understand that, we have to introduce two quantities, which are useful.

两个参数分别称为到达量和边缘量

So, we call them reach and margin.

我们首先要关注的是所谓的封闭分叉

But first observe that we're just gonna focus on what we call a closed fork.

封闭分叉指的是分叉的结尾区块与一个诚实参与方关联

A closed fork is a fork that ends on just nodes that correspond to honest parties.

区块树的叶子节点是一个诚实参与方

So, the leaves of that tree is just honest parties.

注意 我们不需要考虑结尾区块与攻击者关联的分叉

Because we don't need to consider those forks that ends with adversarial blocks,

因为攻击者随时可以在适当的时间在与攻击者关联的分叉后方填充任意区块

exactly because the adversary can always fill them in at the right time.

对于一个封闭区块 我们可以定义两个很有用的参数

Now, for it's one of those times, just observe that we can define two useful quantities.

第一个参数是保留量 其表示攻击者可以在后方增加多少个区块

The first one we call the reserve, which corresponds to how many blocks the adversary can add to extend the certain time.

幻灯片上 上方分叉的保留量是3 因为后面有三个时间槽是属于攻击者的

So, in this particular case, this one has a reserve of 3, because there are 3 adversarial slots coming up.

例如 当诚实参与方在第3时间槽上激活时 攻击者可以保留3个区块 并增加到后方

So, for example, when an honest party is activated here, the adversary can use the reserve of 3 to advance this time by 3 blocks.

同时 还可以定义另一个参数 称为间隔量 表示此分支成为主链所需的时间

At the same time, there is another quantity, which we call gap, which says how far away behind is this time from the leading time of the fork.

如果我们用保留量减去间隔量 我们就定义了所谓的到达量

So, if we take the reserve and we subtract the gap from it, we define a quantity which we call reach.

这个量告诉我们 此分叉需要多长时间才能成为主链

And this says how much this time is behind the leading time in a fork.

在我们的概率分析中 到达量和边缘量是两个非常重要的参数

So, reach and margin are going to be the two important quantities in that probabilistic analysis that we will do.

好消息是 组合这几个参数后我们可以知道 如果一个特征字符串是可分叉的

The nice thing is that, combinatorially you can show that the string is forkable

当且仅当存在一个封闭区块 此区块的边缘量为0 或者至少为0

if and only if there is some close fork for which it holds that the margin is 0, at least 0,

而边缘量指的是我们取最长链的到达量

where the margin is basically you take the time which is the longest,

将最长链的到达量与第二长链的到达量进行对比

and you compare it with the one which is second longest,

并且计算这两个链到达量的差

and you try to see what is the difference between the two.

我们应用到达量和边缘量建立了概率分析模型

So, this is what we show in establishing the probabilistic experiment that determines the reach and margin.

我们会发现 当新的权益拥有者增加区块时 到达量和边缘量构成了随机游走过程

What happens is that reach and margin, they do like a complex random walk as new stakeholders are elected to add the block.

当特征字符为1时 表示此时间槽被攻击 此时到达量和边缘量都增加了1

So, when you have a 1, which means it's adversarial, what happens is that reach and margin are extended by 1.

另一方面 当特征字符为0时 到达量和边缘量都会降低

On the other hand, if you have a 0, what we would like to happen is that reach and margin are decreased.

这一过程反应在幻灯片最下方的这行公式上

Basically, it would have been this setting, just this final line here.

如果事实真的如此 那么到达量和边缘量构成了一个简单的随机游走过程

If it was like that, that would have been a simple random walk.

但实际的协议执行过程中 只有最下方情况到达量和边缘量都会降低

But what happens actually is that this only happens in one case.

在处于最下方情况时 到达量和边缘量都会降低

So, reach and margin decrement in this case.

但实际上 到达量不可能降低到小于0

But what happens is that reach will never drop below 0.

如果到达量是0 则到达量保持为0 因为到达量描述了攻击者的攻击优势

So, if it is 0, it will remain 0, because reach expresses the advantage of the adversary.

这个迭代公式中非常有趣的一点是 如果边缘量为0 则到达量会降低

And now what's very interesting that happens in this recursive formula is that when the margin is 0, then the reach is decremented, which basically means that

因为攻击者可以在牺牲到达量的条件下对边缘量进行补偿

it's possible for the adversary to compensate for the margin for how much it is behind by sacrificing reach.

现在 如果我们应用这个迭代公式 并且将此迭代公式应用在时间槽领导者采样实验中

Now, if you take this recursive formula, and you apply it to the experiment that samples the next slot leader,

我们实际上会得到一个二维随机游走过程

What happens is that we do have a two dimensional random walk.

这个二维随机游走过程实际上非常简单 到达量和边缘量会交替为正或者为负

And these two dimensional random walk is kind of simple, when variable reach and margin are positive and negative respectively.

因此 这两个变量会构成一个简单的随机游走过程

So, then the two variables behave like simple random walk.

但当到达量为正时 边缘量会降低 直到固定为0

But what happens when reaches positive, then margin gravitates and sticks to 0.

因此 这个二维随机游走过程体现出了边缘量和到达量之间的相互关系

So, here's a dependency and the two dimensional aspect of this random walk is revealed.

现在我们要考虑的问题是 当在这个随机游走过程中随机取一个长特征字符串

So now, we have to see what is the probability that if we take a long string in this two-dimensional random walk,

则最终得到边的边缘量至少为0的概率是多少

we're going to end up with a margin which is at least 0.

我们可以证明 这一情况出现的概率非常小 等于2^-√n乘以一个常数

And what we prove is that this will happen with very small probability, that's going to be 2^-√n times a constant.

关键点在于 我们可以把二维空间划分为3个区域 分别称为热区、脆弱区、冷区

And the key idea is that we can divide the two-dimensional space into three areas, what we call them hot, volatile, and cold.

脆弱区指的是边缘量和到达量的取值都接近为0

What happens is that the volatile area is the one which is at the center of it of the two axis when the margin and reach are close to 0.

这实际上就是初始化状态

And that's actually our initial state.

我们还有冷区 此时边缘值为负 此种情况下区块链是安全的

And then, we're going to have a cold region, which basically when the margin is negative. And that's safe.

我们还有热区 此时区块链是不安全的

And then we're going to have a hot region, where basically we've lost the game.

我们可以证明 在游戏开始时我们的初始状态为脆弱状态

So, what happens in what we prove is that if you start from the volatile state as it is the case in our game,

从脆弱状态转移到热状态的概率无限接近于0 为2^-√n

the probability of moving to the hot state is negatively close to 0, 2^-√n,

但从脆弱状态转移到冷状态的概率为常数

whereas there is a concept probability of moving to cold.

核心观察结论在于 一旦转移到了冷状态 则区块链将永远为冷状态

And the key observation is that once you move to cold, then you will remain to cold.

也就是说 随机游走过程的位置已经远离脆弱区和热区 我们是安全的

Basically random walk has moved away from the volatile and hot territory. And now we are safe.

由于转移到热区的概率是2^-√n 而转移到冷区的概率是常数

Given that all this happens with error 2^-√n, and we have a concept probability of doing this transition,

我们要做的是把时间划分成√n个区域 证明在任意一个处理过程中

what we need to do is divide this into √n regions, and then we can see that with very high probability,

区块链都会有很高的概率从脆弱状态转移到冷状态

in one of those steps of the process, you're going to transition from the volatile to cold state.

这意味着我们是安全的

And basically that means that you are safe.

最后 这意味着获得一个可分叉特征字符串的概率也是可忽略的

So, finally, this concludes that the density of this forkable string is just that.

有了这样一个基础协议 我们可以进一步设计出权益动态变化下的协议

So, armed with this, now we show how we can actually go to the dynamic state case.

在权益动态变化的条件下 我们要做的是在特定数量的时间槽中执行基础协议

The dynamics state case, what happens is that we run an instance of the basic protocol for a certain number of slots,

当然 我们要假定创始区块初始化了一个随机种子

assuming some initial randomness seed which is part of the genesis block.

现在 假定我们有一个神奇的锚点 当区块链生成到给定的锚点时

And now, let's assume that we have a magic beacon, that when we reach that end of that sequence of blocks,

我们可以得到一个新的种子 从而重新根据权益拥有者的权益分布生成新的记账顺序

it will give us a new seed that we can use to sample again from the stakeholder distribution.

到达锚点后 锚点会给我们另一个种子 我们用这个种子重新分配记账顺序

So the beacon comes, it gives us another seed, and that's what we need about this.

我们需要利用区块链自身的特性来更新权益拥有者的记账概率分布

We can use the blockchain itself to refresh our stakeholder distribution,

而新的记账概率分布由一系列交易执行过程后 各个参与方的新权益量所决定

and use the new stakeholder distribution that has arisen as the stake has shifted on this initial sequence of blocks.

也就是说 我们根据区块链的一部分记录结果 应用锚点重新生成种子

So here, you can see this self-referential aspect that we use the blockchain that we have here, the short segment, and we can reseed that process using the beacon, choosing a new stakeholder distribution.

然后再用新的种子创建更新后的权益拥有者记账概率分布

Now, this is used now with a new seed to create another stakeholder distribution.

协议继续执行下去

And the protocol continues.

当然了 在现实生活中我们没有这样一个可信锚点

So, of course, in practice, we do not have such a trusted beacon.

这么解决这个问题呢？

So, how can we solve that?

答案是利用G.O.D 即保证“产出交付的硬币投掷”

And the answer is coming from G.O.D, or more specifically, guaranteed output delivery coin tossing.

“保证产出交付的硬币投掷” 或称G.O.D硬币投掷 可以确保输出的随机性

So, guaranteed output delivery coin tossing, or G.O.D coin tossing, has this godly property that it guarantees the output.

其基本意思是 如果协议的大多数执行方都是诚实的 则保证输出结果是无偏随机种子

Basically that means that if you have honest majority, you are guaranteed to produce an unbiased seed.

我们要做的是要执行一小段时间的基础协议 产生一部分区块片段

So, what we have to do is use this little short opportunity of time we have to create this blocks in segments

这样我们就可以执行G.O.D硬币投掷协议 从而为下一个纪元产出一个随机种子

so that we can run this G.O.D coin tossing protocol that is going to be enough to produce the seed for the next epoch.

整个协议执行过程需要应用一个称为公开可验证密钥协商协议的工具

And that is going to be done by using this tool called publicly verifiable secret sharing,

而我们可以通过简单的密码学工具构建公开可验证密钥协商协议

which is something that we can construct using simple cryptographic tools.

应用上述提到的所有工具和协议 我们最终证明账本可以达到目标

So, based on all that, we finally have our proof ledger objective.

虽然我没有时间讲解整个证明的细节 但我们可以通过理论证明

And even though I don't have the time to go over the detail of the proof, what we show is that

假定攻击者具有一定程度的适应性攻击能力

The Ouroboros protocol, assuming a certain restriction on how adaptive is the adversary,

则Ouroboros协议的公共前缀、链质量、链生长速度等的错误率都是k的可忽略函数

we can prove the properties of common prefix, chain quality, and chain growth that you've heard in previous talks with negligible error in k,

这意味着可以用Ouroboros实现一个健壮的交易账本

which means that Ouroboros will give rise to a robust transaction ledger.

我在本次讲座中没时间讲解论文中的全部内容了 各位可以在论文中发现一些其它成果

So, some things I don't have the time to cover in this talk, but you can find in the paper is that

除了上述协议之外 我们还设计出了一个基于权重的激励模型

on top of what we have, we also provide an incentive structure for the protocol based on weights.

我这里跳过这几页幻灯片

And I'm going to skip those slides now.

我们直接来看最后的结论吧

But I'm just going to focus on this final point.

我们证明 在Ouroboros协议上执行所设计的激励奖励机制可达到近似纳什均衡

We actually prove that for this incentive reward mechanism that we have, we can prove that running the Ouroboros protocol is an approximate Nash equilibrium.

而且与比特币相比 我们的激励机制与协议是完全兼容的

And the good feature about this is that contrary to Bitcoin, we can actually establish this incentive compatibility of the protocol.

进一步 我们设计实验测试了协议的性能 具体结论在这页幻灯片上

Some further details, and I'm gonna finish with some performance characteristics that you can see here.

我们在亚马逊云上用40个节点运行了实验

This is an implementation using 40 nodes in the Amazon Cloud.

我们实现了我们的协议 并测试了协议的性能

We have implemented our protocol and we've benchmarked it.

大家可以看到 权益证明协议可以达到良好的性能

And you can see here appreciate the efficiency of the proof of stake protocol.

在这种简单的协议实现下 每秒确认的交易数量可以达到250个

In this case, the transaction per second is about 250 in this simple implementation that we have.

大家可以把这个结果和每秒确认7、8、或者9个的比特币协议进行对比

You can compare this with 7, 8, or 9 transactions per second that you might get from a protocol like Bitcoin.

后续工作中 我们将解决本工作尚没有解决的问题

So, some upcoming work that we have is solving two open questions that were left in this one.

后续工作将解决半异步、或者部分同步网络 攻击者合谋条件下的协议设计问题

So, semi-asynchronous setting, or basically partial synchrony, and adaptive corruption is dealt with in an upcoming work

这个工作是我和Bernardo David、Peter Gazi和Alexander Russell一起完成的

joined with Bernardo David, Peter Gazi, and Alexander Russell.

大家可以在ePrint上读到此篇论文

And you can find this on ePrint.

这些也是后续工作 大家在网上可以找到演讲视频

So, further work is here. And you can find the talk online.

讲座时间快到了 这些是相关工作 大家可以在ePrint版本或会议版本找到相关文献

And time is running out, but this is related work, which you can find a comparison in the ePrint version of the paper as well as in the proceedings.

这就是幻灯片的最后一页了

And this brings me to the end.

感谢大家前来聆听讲座

Thank you very much for your attention.